CLAIMS

What is claimed is:

1. A system, comprising:

a processor executing an application;

a peripheral device coupled to the processor;

memory containing an application data structure accessible by said application, wherein accesses to said application data structure and accesses to said device are formatted differently, and wherein data can be written to, or read from, the peripheral device via the application data structure; and

reformat logic coupled to the processor and memory, the reformat logic dynamically reformats an access from the application targeting the application data structure to a format that comports with the device, thereby permitting said application to manage the peripheral device without the use of a device driver.

2. The system of claim 1 wherein the peripheral device comprises a display.

3. The system of claim 1 wherein the application data structure comprises an array.

4. The system of claim 3 wherein the array comprises a multi-dimensional array.

5. The system of claim 3 wherein array comprises a single-dimensional array.

6. The system of claim 1 further including a device buffer associated with the device and wherein the application data structure comprises an *n*-bit data structure and the device buffer comprises an *m*-bit display buffer, wherein *n* is different than *m*, and the reformat logic reformats an *n*-bit access from the application to an *m*-bit access for the device buffer.

7. The system of claim 6 wherein *n* is not an integer multiple of *m* and the reformat logic includes alignment logic to implement a read-modify-write operation to write a value from the application data structure across byte boundaries in the display buffer.

8. The system of claim 6 wherein *m* is less than *n*.

9. The system of claim 6 wherein the reformat logic comprises a plurality of registers which are programmable to store a plurality of values, said values comprising information that is indicative of the starting and ending addresses of the application data structure in which accesses are to be reformatted by the reformat logic, the starting address of the device buffer, *n*, and *m*.

10. The system of claim 6 wherein the reformat logic comprises a plurality of registers which are programmable to store a plurality of values, said values comprising information that is indicative of the starting and ending addresses of the application data structure in which accesses are to be reformatted by the reformat logic, the starting address of the device buffer, *n*, and value indicative of the ratio between *n* and *m*.

11. The system of claim 6 wherein the registers are programmed by a virtual machine.

12. The system of claim 1 further comprising a multiplexer that selectively permits accesses from the application to be provided to the memory without being reformatted by the reformat logic and permits accesses from the application to be reformatted before being provided to the memory.

13. The system of claim 12 wherein the reformat logic controls the multiplexer to select whether or not a reformatted access is to be provided to the memory.

14. The system of claim 12 wherein the processor supplies an address to the multiplexer and to the reformat logic and asserts a signal to the multiplexer to cause the multiplexer to select whether or not a reformatted access is to be provided to the memory.

15. Reformat logic, comprising:

a plurality of registers; and

translation logic that accesses the registers and that receives a memory access targeting an application data structure that has a different format than for accesses that are provided to a device that is external to said reformat logic and that reformats the request to a format compatible with the device based on values stored in the registers.

16. The reformat logic of claim 15 wherein the device has an associated memory buffer and wherein the application data structure is *n*-bit accessible and the memory buffer is *m*-bit

accessible where *m* is less than *n*, and the translation logic reformats the request from an *n*-bit format to an *m*-bit format.

17. The reformat logic of claim 16 wherein *n* is not an integer multiple of *m* and the reformat logic includes alignment logic to implement a read-modify-write operation to write a value from the application data structure across byte boundaries in the display buffer.

18. The reformat logic of claim 16 further comprising a plurality of registers which are configured to be programmed to store a plurality of values, said values comprising values that enable to determine the starting and ending addresses of the application data structure in which accesses are to be reformatted by the reformat logic, the starting address of the memory buffer, *n*, and *m*.

19. The reformat logic of claim 16 further comprising a plurality of registers which are configured to be programmed to store a plurality of values, said values comprising values that enable to determine the starting and ending addresses of the application data structure in which accesses are to be reformatted by the reformat logic, the starting address of the memory buffer, *n*, and value indicative of the ratio between *n* and *m*.

20. The reformat logic of claim 15 wherein the translation logic reformats both read and write requests.

21. The reformat logic of claim 15 wherein the memory buffer for which the reformat logic reformats a request comprises a display memory buffer associated with a display.

22. A method, comprising:

receiving a physical address from a processor, the physical address associated with an application data structure;

converting the physical address to a device buffer address associated with a device buffer, the device buffer being accessed with a different number of bits than the application data structure; and

providing the converted device buffer address to the device buffer to permit the processor to control a peripheral device without using a driver associated with the peripheral device.

23. The method of claim 22 wherein receiving the physical address from the processor is performed upon writing to the application data structure.

24. The method of claim 22 wherein receiving the physical address from the processor is performed upon reading from the application data structure.

25. The method of claim 22 further including completing a read or write transaction to the device buffer using the converted device buffer address.

26. The method of claim 22 wherein the device buffer is accessed with fewer bits than the application data structure.